TÉCNICO LISBOA

Machine Learning

Project Report

Authors:

Duarte Sá Morais (100163) duarte.morais@tecnico.ulisboa.pt
João Dias (100202) joao.v.dias@tecnico.ulisboa.pt

Group 10

1 Regression

1.1 First delivery

1.1.1 Regression Models

The primary objective of this analysis was to make predictions based on a dataset consisting of 15 examples, each with 10 distinct features. The analysis involved the assessment of seven distinct regression models, including Linear Regression, Ridge Regression, Lasso Regression, Lasso LARS, Elastic Net, and Orthogonal Matching Pursuit (OMP), in terms of their predictive capabilities. To evaluate the performance of these models, a ten-fold cross-validation (k = 10) approach was adopted. In this method, the training data was divided into ten subsets, with nine of these used iteratively for model training, and one held out for validation. The performance of these models was assessed using the Mean Squared Error (MSE) metric.

Table 1: Comparison of Regression Models

Model	**Mean of MSE**	**Best hyperparameter**
Linear Regression	9.422	-
Ridge Regression	3.154	$\lambda = 2.40$
Lasso Regression	3.384	$\lambda = 0.10$, zero coefficients = 4
Lasso LARS	3.368	$\lambda = 0.09$, zero coefficients = 6
Elastic Net	3.340	$\gamma = 0.1, \lambda = 0.4$
Orthogonal Matching Pursuit	2.680	zero coefficients=5

1.1.2 Model Selection and Performance

The table in this section provides an overview of the MSE scores and optimal hyperparameters for each model, allowing for a thorough comparison of their effectiveness.

Surprisingly, the OMP model emerged as the optimal selection, demonstrating the lowest MSE at 2.680. OMP's natural ability to induce sparsity, much like Lasso and other methods, enabled it to perform exceptionally well in a dataset with high dimensionality. It accomplished this by identifying only five non-zero coefficients, highlighting its significance in feature selection and efficient modeling.

1.1.3 Data Analysis

In the course of the initial data analysis, it became apparent that the dataset's features exhibited different degrees of importance. This observation was reinforced by the subsequent OMP analysis, which revealed that only five out of the ten features possessed non-zero coefficients, emphasizing the need for a feature selection model. OMP is a greedy method for feature selection, iteratively selecting the most relevant features, while Lasso is a regularization technique that encourages automatic feature selection by shrinking some feature coefficients to zero.

It's worth noting that Lasso Regression performed less effectively than Ridge Regression in this study, as evidenced by a higher MSE. This might be attributed to the removal of too many features (resulting in excessive zero coefficients) or the use of a smaller λ value in the Lasso approach.

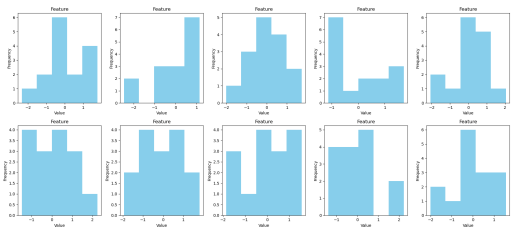

Figure 1: Dataset with normalization

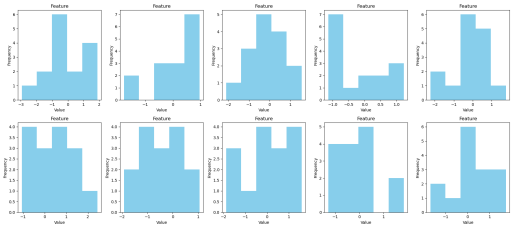

Figure 2: Dataset without normalization

In this analysis, an important consideration was made regarding data normalization. The decision not to normalize the data was based on the visualization of the histograms in figures 6 and 2, where it became evident that normalization was redundant. Furthermore, the normalization of the data was found to result in worse results.

1.2 Second delivery

In the second task, the primary goal remained consistent with the previous one: making predictions using a dataset generated by two different linear models. This dataset included 100 examples, each with four unique features, and the challenge was to distinguish between instances created by these two distinct models without prior knowledge of their origins.

1.2.1 Data preprocessing

The initial analysis steps closely followed those of the previous task. Data preprocessing steps were unnecessary as the data distributions were already normalized, evident from the histograms (figure 3 and 4).

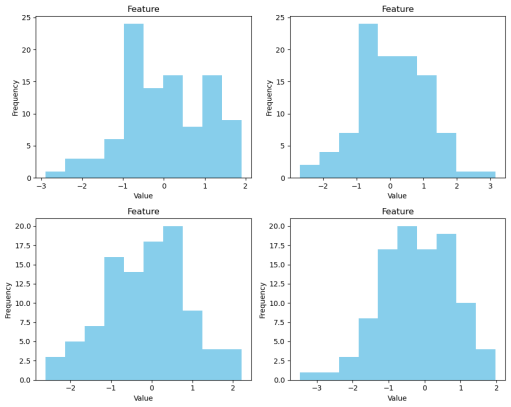

Figure 3: Input train data distribution

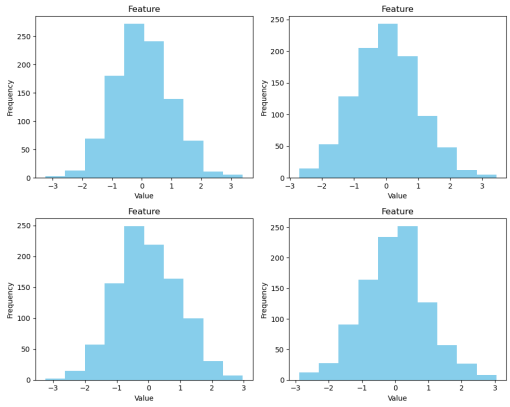

Figure 4: Input test data distribution

1.2.2 Clustering models

In this task, clustering techniques were introduced. Specifically, K-Means and the Gaussian Mixture Model (GMM) were applied. These techniques effectively divided the training data into two distinct clusters, simplifying the process of creating individual regression models. These models were fine-tuned to align with the unique characteristics of the underlying generative models.

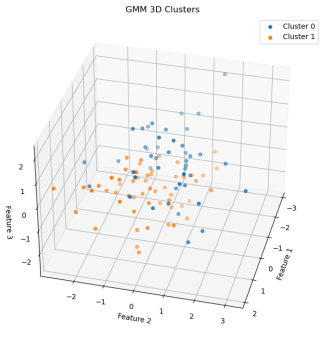

Figure 5: Pointcloud from 3 features

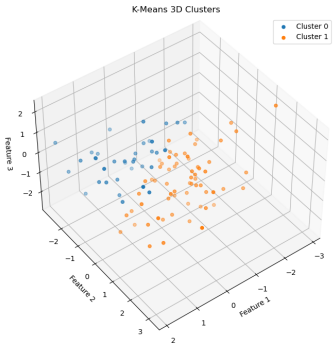

Figure 6: Pointcloud from 3 features

Furthermore, when looking at Figure 3 and 4 is evident that both the test and training datasets displayed Gaussian distributions. Consequently, the Gaussian Mixture Model emerged as the primary choice for the initial clustering approach, while also considering the K-Means method.

In the context of the analysis performed, various approaches were employed to select the most suitable regression models for predicting outcomes in a dataset generated by two different linear models.

1.2.3 Residuals approach

One approach involved residual analysis, where a linear regression model was initially applied to the training data to predict the target variable. Subsequently, residuals were calculated,

Table 2: Summary of Approaches and MSE

Approach	Folds	MSE	Best Models
K-Means (Residuals)	2	1.42	Model 1: OMP (0.79) Model 2: Ridge (2.05)
GMM (Residuals)	2	1.45	Model 1: Ridge (0.30) Model 2: Ridge (2.60)
K-Means (Input)	2	1.20	Model 1: Ridge (1.47) Model 2: Elastic (0.93)
GMM (Input)	3	1.27	Model 1: OMP (1.15) Model 2: Ridge (1.38)
GMM (Input and Output data)	3	0.05	Model 1: Ridge (0.06) Model 2: OMP (0.04)
K-Means (Input and Output data)	4	0.56	Model 1: Ridge (0.53) Model 2: OMP (0.59)

and a clustering method was applied to divide the data into two regression models. However, it was recognized that this approach might not be ideal, as it assumed that the residuals adhered to a normal distribution, which was not guaranteed. A counter example of this method would be if the distributions were like a "cross", not making it viable.

1.2.4 Input training data approach

Another approach was focused on clustering using only the input training data. This approach presented the challenge of not considering the output training data for clustering, leading to an inaccurate data division.

1.2.5 Input and output training data approach

Consequently, the final approach integrated clustering with all training data, both input and output training data, followed by the application of various regression models for both datasets derived from clustering.

Because the data's distribution looked a lot like a normal distribution, naturally favored the Gaussian Mixture Model (GMM) as the standout performer, which creates clusters of the same size and density. Also, the GMM is more flexible in contrast to K-Means clustering, which rigidly forms equally sized and spherical clusters with similar densities. It could handle elliptical clusters of varying sizes and densities, aligning more closely with the actual data distribution.

Ultimately, the model that delivered the most outstanding results was Ridge Regression, coupled with prior clustering using the Gaussian Mixture Model. This combination achieved the lowest MSE of 0.046, highlighting the success of this approach in the analysis.

2 Classification

2.1 First delivery

The second part of this project was aimed at creating an image classifier for dermoscopy images that would receive a 28x28 image with 3 color channels and define it's class. The first

delivery was a binary classification problem, where the model should classify the image into either a nevu or a melanoma.

Since there are a lot of classifiers to be taken into account, such as multilayer perceptrons, convolutional neural networks, Naive Bayes classifier, linear classifiers, support vector machines, decision trees ... and the project was a time sensitive problem, it was decided to consider two different classifiers and try to optimize them. Finally, the classifiers that were used was a convolutional neural network and SVM, with special focus in the CNN due to their great use in image classification. Due to this, the following sections will correspond to the CNN model and the SVM will be explained after.

2.1.1 Data Analysis

The first steps to create an image classifier is to understand what type of input is the classifier expecting and to do some data preparation to ensure that the classifier will get the best possible results.

Having this in mind, the *numpy* input vectors corresponding to Xtrain and Xtest were divided by 255, the maximum value of each vector. Since these vectors represent pixel information, their values range from 0 to 255, corresponding to darker and brighter pixels. By dividing the vectors by the maximum value in them, the data is scaled down to the interval [0,1] which leads to better results in the neural network.

Once this was done, the X vectors were shaped into a 4D vector (number of images x 28 x 28 x 3) corresponding to the image format. This allows the user to plot images and is the defined input for the CNN. Also, it should be noted that the y vector was transformed into a one-hot enconding vector of 2 columns, since there are 2 classes. When inspecting this matrix, it was possible to confirm what was already stated in the project paper regarding the presence of data imbalance, with one class having, approximately, 7 times more instances.

Finally, to reduce bias in the model, the vector composed by the training data and corresponding labels were shuffled and then split into training and validation data, in order to have a way to evaluate the network during training. It was decided that 15% of Xtrain would be validation data.

2.1.2 Creating and training model

For the creation of the neural network, it was considered different architectures (ranging from simpler to more complex), as well as different approaches to tackle the imbalance problem, such as adding class weights to incentivize picking the minority class during training, and using data augmentation techniques SMOTE and keras' Image Data Generator, which generates new images from existing ones, rotating, shifting horizontally or vertically, flipping and zooming them.

In order to evaluate the model after each epoch, an early stopping callback was implemented with patience equal to 15 which stopped the fitting process if the loss function did not get better within a 15 epochs radius.

Regarding the architectures of the convolutional networks, sequential networks were created with Convolutional, MaxPooling, Fully Connected and Dropout layers.

The outer convolutional networks always had less parameters with the objective of captivating the more general features, while the deeper convolutional layers had a higher number of

Table 3: Results obtained for just the input data and class weights approach

Approach	Just input data				Class weights			
Metric	F1 - C0	F1 - C1	BAcc-T	BAcc-Val	F1 - C0	F1 - C1	BAcc-T	BAcc-Val
CNN1	0.9348	0.1085	0.9206	0.6702	0.8337	0.4173	0.6365	0.6349
CNN2	0.9377	0.25	0.9244	0.7209	0.7392	0.3784	0.6703	0.6476
CNN3	0.93887	0.2639	0.9231	0.7001	0.8256	0.4282	0.6441	0.6305

Table 4: Results obtained for SMOTE and Balanced Augmentation

Approach	SMOTE				Balanced Augmentation			
Metric	F1-C0	F1-C1	BAcc-T	BAcc-Val	F1-C0	F1-C1	BAcc-T	BAcc-Val
CNN1	0.7791	0.4481	0.8062	0.6277	0.9310	0.4549	0.9320	0.7558
CNN2	0.8353	0.4193	0.8155	0.6276	0.9372	0.4878	0.9348	0.7352
CNN3	0.808	0.4403	0.8161	0.6376	0.9293	0.5950	0.9444	0.7855

filters to "see" and "understand" the more complex features. It should also be noted that all convolutional networks were created with padding that creates a border of zero-valued pixels for each image with the objective of keeping the same resolution without adding noise. MaxPooling and Dropout layers were also used as a regularization technique in order to reduce the possibility of overfit. MaxPooling would pick the maximum number of a pool of 4 values and output to the next layer, while Dropout "drops" a defined amount of weights from a layer to the other.

Since the evaluation metric used by the teaching team was Balanced Accuracy, custom functions that evaluated the specificity, sensitivity and balanced accuracy were created to track how the network was behaving during training. The results obtained for the best network and for each approach are summed up in tables 3 and 4.

2.1.3 Choosing a model

In order to decide what classifier was best for the task in hand, it was taken into account the balanced accuracy score, the F1 score and the way the loss value for training and validation data evolved for each epoch.

Looking at tables 3 and 4 it can be easily seen that using the oversampling technique Balanced Augmentation resulted in better results than class weights and SMOTE, which led to trying to optimize the models which use kera's Data Generator instead of improving the others. The poor performance of class weights is due to the fact the the weight of the minority class was too high, leading to the classifier to over-predict for this class.

Having this in mind, the best convolutional neural network obtained was CNN3 using balanced data augmentation, and it was the final method used to predict the test data.

2.1.4 Results discussion

Even though the BAcc results present in table 4 for the chosen model were promising, this group classification for this task wasn't as great, achieving only a BAcc of 0.7341 for the test data in domain and 0.5781 for the test data out of domain.

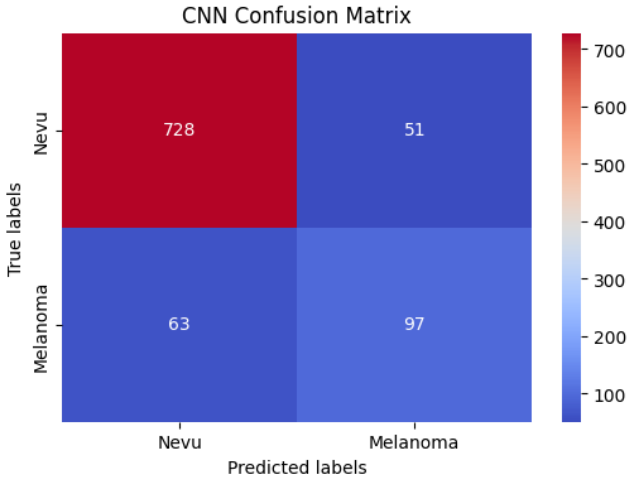

Figure 7: Confusion matrix for CNN3 using validation data

Observing the confusion matrix using validation data for CNN3 with balanced augmentation (in figure 7), it can be seen that this was expected. For 143 melanoma images, the neural network only correctly predicted the label 60.6% of times, a poor result, even though that extra images for this class were created. Unfortunately, the confusion matrix was only plotted after the delivery deadline already passed, which is a big mistake since it can represent important information to access the performance of a classification model.

This is due to the fact that the network was overfitted for the training data and, despite that data oversampling was done, the training data has way more "real" instances of nevus, since the majority of training images used for melanomas were synthetic and didn't fully correspond to real images.

The evidence of an overfitted network can also be seen when computing the mean value of F1 score and observing the drop off from training to validation data (going from 0.9277 to 0.6846) that should have been paid more attention to. This is because F1 score is a metric that, as explained in 2.2.2, accounts for both precision (how many of positive predictions are correct) and recall (how many of the actual positive labels were predicted correctly), being a great metric to evaluate classifier models trained on imbalanced data because the final result accounts for each class in question, but since the balanced accuracy result achieved was almost of 80%, it was thought that the network wasn't overfitted and had good results.

Another red flag that should have been accounted for when solving this assignment is the low F1 score for class 1 (melanoma), as seen in 3 and 4. Even though the final network had the highest value of this metric, it was only 0.5950 that, when contrasting with the F1 score of the other class (0.9293), shows that this model wasn't as good as previously thought.

The main reason for this network behaving this way is due to the fact that it's architecture is too complex for the binary task in hand, having too many weight/parameters that lead to overfitting, even though that Dropout layers were used to prevent this issue. This was taken into account in 2.2 where simpler networks were created, with less number of layers, filters and especially fully connected layers (where the majority of the weights are present).

Going back to the result obtained from the teaching team in this assignment, it's worthy to underline the fact that the test data that had the same origin as the training data got better

results than the one out of the domain, another indicator of the present overfit regarding the training data in the current model.

Finally, and even though the obtained results were poor, when comparing the value of just using input data (3) with the results of Balanced Augmentation (4), it can be seen an improvement in F1 of C1 and BAcc of validation data, showing that the approach to tackle data imbalanced was correct, but badly implemented. It should also be noted that, for the second classification task, it was used a different way to compute BAcc using a library defined function instead of a custom metric to make sure that the obtained score hasn't any errors in it's computation.

2.1.5 Support Vector Classifier

As it was said before, a bigger focus was given to the development of the neural network due to the fact that the convolutional layers do a great job in picking important image information to classify them. This being said, a SVM approach was also explored using a Support Vector Classifier from the library SK learn.

When picking a SVC model there are three main parameters that should be defined: the kernel used in the algorithm, gamma (kernel coefficient) and C (regularization parameter).

For the kernel three options were considered: *linear*, *rbf* and *poly*. The first is used for simpler models where the data is linearly separated, since it was not know this was the case, it wasn't used. *Poly* is used if the image data follows polynomial relations, while *rbf* is a Gaussian kernel that is more general, which led to it's choice. The gamma and C values were defined as [0.0001, 0.001, 0.1, 1] and [0.1, 1, 10, 100], respectively, and a Grid Search was executed to pick the optimal parameters. It was obtained a BAcc value of 0.5549, F1-C0 of 0.92 and a F1-C1 of 0.20.

The low value of balanced accuracy, as well as the F1 score, gives important information regarding the model's incapacity to properly predict 'melanoma' images, probably being overfitted. Since Grid Search is a time consuming process to tune parameters, it was decided to shift the focus of this task to creating a working CNN.

2.2 Second delivery

In this study, we focused on the classification of 2D medical images from two distinct datasets: dermoscopy and blood cell microscopy. Our objective was to classify these images into six different categories, namely nevus (0), melanoma (1), vascular lesions (2), granulocytes (3), basophils (4), and lymphocytes (5). The input images were of dimensions 28x28x3 and were presented as vectors. The primary metric for evaluating the performance of our models was balance accuracy, f1 score and also the analysis of the confusion matrix.

It should be noted that in the creation of the solution to this task a similar approach as described in 2.1.1 was used.

2.2.1 Model Selection

Several Convolutional Neural Network (CNN) architectures were explored to capture the intricate patterns within the medical images. Firstly, a similar train of thought as the one

Table 5: Summary of Approaches and BAcc and F1scores for Model 1

	BAcc	Mean F1score (Train)	Mean F1score (Test)	F1 Score for Each Class
Imbalanced Data	0.64	0.56	0.57	[0.92 0.17 0.42 0.86 0.76 0.85]
Class Weights	0.76	0.59	0.56	[0.77 0.41 0.35 0.78 0.69 0.78]
Data Augmentation	0.72	0.62	0.63	[0.94 0.32 0.63 0.88 0.75 0.86]
Random Oversampling	0.82	0.84	0.66	[0.86 0.53 0.45 0.85 0.80 0.86]

explained in 2.1.2 for CNN architecture was followed, creating simpler networks to avoid overfitting. In the end, the network that had better results and was chosen had more filters in the outer convolution layers than the inner layers which goes against what was expected. This means that the network starts by filtering more "niche" features first and adding less weights along the way leading to less overfit.

To address the class imbalance issue in the dataset, a random oversampling technique was employed for the minority classes, ensuring a more balanced representation of different categories by randomly selecting and duplicating samples from the minority classes.

The class weights approach was then applied to mitigate the imbalance issue, where higher weights were assigned to the minority class during model training. This made the algorithm pay more attention to the minority class and prevented it from being overwhelmed by the majority class.

Lastly, data augmentation techniques were experimented with to increase the diversity and robustness of the models, making the model less likely to overfit. This involved applying rotation, horizontal and vertical shifts, shearing, zooming, and horizontal flipping to artificially expand the dataset.

2.2.2 Results discussion

Our experiments led to some insights. When the CNN models were trained using oversampling, class weights, or data augmentation, it was observed a significant improvement in their performance in classifying images.

Furthermore, based on the analysis of the results from both CNN models, as shown in Tables 5 and 6 it is evident that model 2 consistently outperformed Model 1. It's important to highlight that all the approaches used to mitigate data imbalance issues led to better results.

For both models, random oversampling achieved the best results when compared to data augmentation and the class weights approach. This success can be explained due the fact that data augmentation can change the original data in ways that compromise its integrity and structure. In contrast, oversampling simply adds more instances of the minority class while preserving the existing information. As for the class weights approach, it sometimes leads to biased results by overemphasizing the minority class. These results highlight the importance of selecting the right method to address class imbalance, with random oversampling proved to be highly effective for this problem.

Although model 2 displayed the same balanced accuracy (BAcc) for class weights and random oversampling, the choice was made in favor of random oversampling due to the higher F1 score for the six classes, which stood at 0.84.

In the previous tasks, the primary priority had been placed on BAcc as the key metric,

Table 6: Summary of Approaches and BAcc and F1scores for Model 2

	BAcc	Mean F1score (Train)	Mean F1score (Test)	F1 Score for Each Class
Imbalanced Data	0.73	0.80	0.72	[0.93 0.51 0.48 0.94 0.90 0.92]
Class Weights	0.84	0.83	0.79	[0.86 0.52 0.50 0.93 0.90 0.90]
Data Augmentation	0.75	0.76	0.63	[0.93 0.54 0.61 0.86 0.75 0.77]
Random Oversampling	0.84	0.96	0.82	[0.92 0.63 0.69 0.96 0.94 0.95]

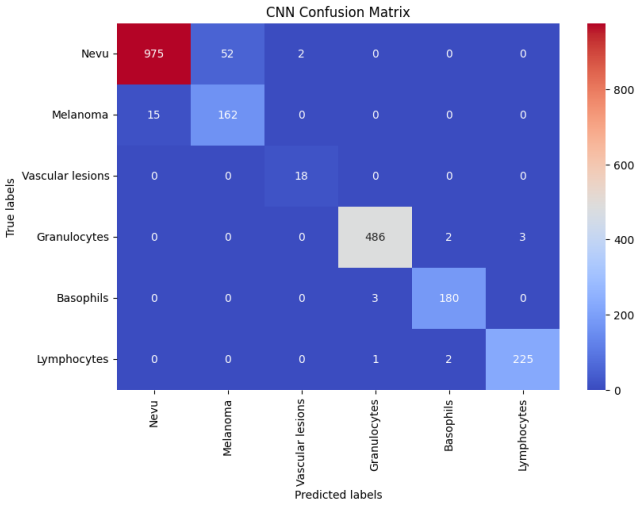

Figure 8: Confusion matrix for second delivery for validation data

resulting in suboptimal evaluations. This was due to the fact that BAcc treats all classes equally and does not adequately assess the model's ability to correctly classify the minority class. As a result, it became clear that the F1 score plays a crucial role in the context of imbalanced data. For that reason, it was prioritized in the analysis because it provides a comprehensive evaluation of a model's performance by considering both false positives and false negatives. This makes the F1 score a robust metric, especially when some classes are underrepresented. In the analyses perform, it was also examined the confusion matrix for all cases presented in Tables 5 and 6. This allowed to identify true positives, true negatives, false positives, and false negatives, amplifying our assessment by providing a detailed breakdown of classification outcomes, thereby complementing the balanced accuracy (BAcc) and F1 score metrics.

In the figure 8 it shown the confusion matrix for the approach with the best results, where the lowest correct probability is 91.5% for 'melanoma', a big improvement from figure 7. This means that not only changing the data augmentation approach to over sampling, as well as the changes made to the created neural networks, had a great impact in the obtained results.